南京大学
NANJING UNIVERSITY

高级机器学习

强化学习
(reinforcement learning)

回顾：强化学习的关键要素

- Agent
- Environment
- State s
- Action a
- Reward r
- Policy $\pi(s, a)$
- State transition $p_{sa}(s')$

- Return:

$$G_t = R_t + \gamma R_{t+1} + \gamma^2 R_{t+2} + \cdots$$

- Action-value function:

$$Q_\pi(s_t, a_t) = \mathbb{E}\left[G_t | s_t, a_t\right].$$

- State-value function:

$$V_\pi(s_t) = \mathbb{E}_{a\sim\pi}[Q_\pi(s_t, a)].$$

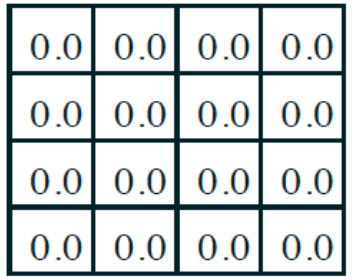
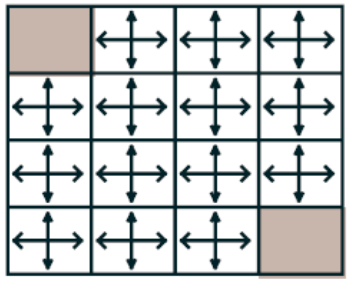
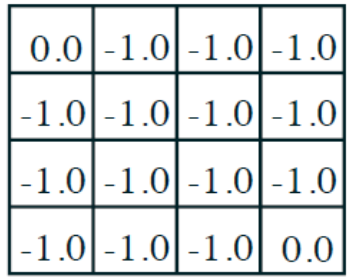
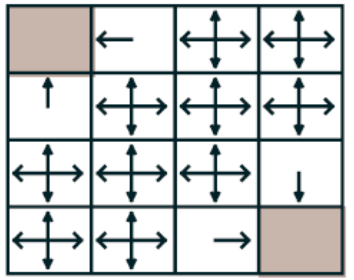
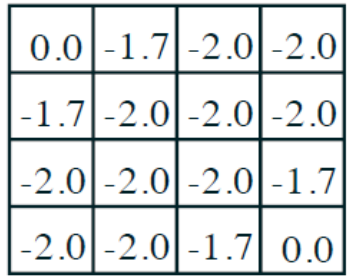
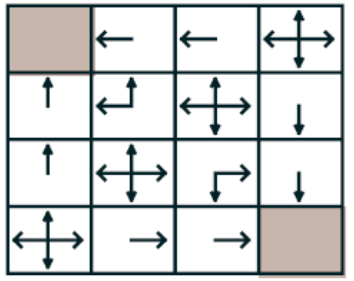
回顾：有模型学习

- 策略评估：bellman等式

$$V^{\pi}(s) = \mathbb{E}[r_1 + \underbrace{\gamma r_2 + \gamma^2 r_3 + \cdots}_{\gamma V^{\pi}(s_{t+1})} | s_0 = s]$$

$$= \sum_{a} \pi(s,a) \sum_{s' \in S} P_{sa}(s') \left(r_{sa}(s') + \gamma V^{\pi}(s') \right)$$

- 价值迭代
- 策略迭代

	随机策略的V_k	V_k对应的贪心策略
K=0		
K=1		
K=2		

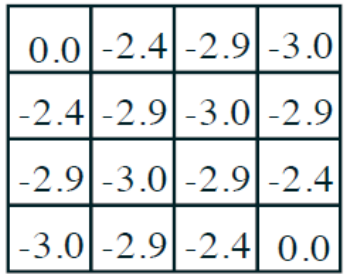
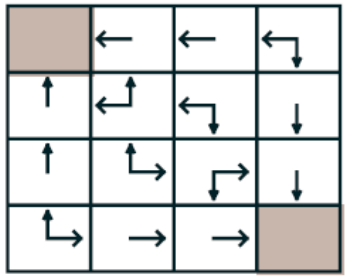
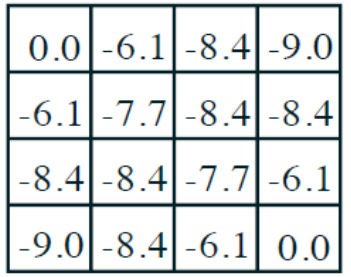
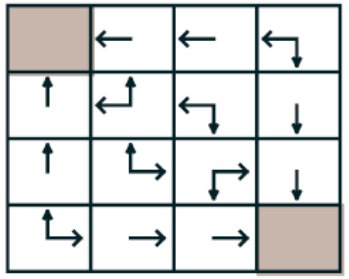
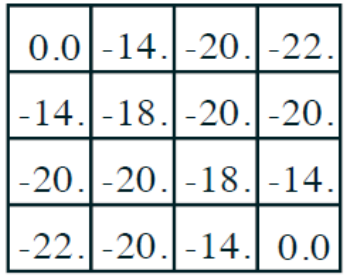
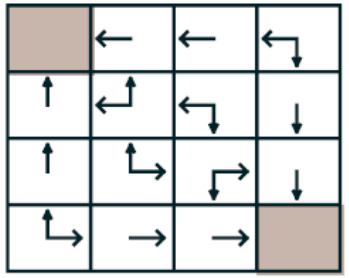
回顾：有模型学习

- 策略评估：bellman等式

$$V^{\pi}(s) = \mathbb{E}[r_1 + \underbrace{\gamma r_2 + \gamma^2 r_3 + \cdots}_{\gamma V^{\pi}(s_{t+1})} | s_0 = s]$$

$$= \sum_{a} \pi(s,a) \sum_{s' \in S} P_{sa}(s') \left(r_{sa}(s') + \gamma V^{\pi}(s') \right)$$

- 价值迭代
- 策略迭代

	随机策略的V_k	V_k对应的贪心策略
K=3		
K=10		
K=∞		

回顾：有模型学习

- 有模型学习小结
  - 强化学习任务可归结为基于动态规划的寻优问题
  - 与监督学习不同，这里并未涉及到泛化能力，而是为每一个状态找到最好的动作

- 问题：如果模型未知呢？
  - 从“经验”中学习一个MDP模型
  - 不学习MDP，从经验中直接学习价值函数和策略：
    - 模型无关的强化学习（Model-free Reinforcement Learning）

大纲

- 强化学习简介
- 多臂老虎机
- 有模型学习
- 免模型学习
- 深度强化学习

价值学习与策略学习

- 价值学习（Value-based learning）通常是指学习最优价值函数$Q^*(s,a)$，假设我们有了Q^*，智能体就可以根据Q^*来做决策，选出最好的动作

- 例如，以超级马里奥为例，每次观测到一个状态s_t，让Q^*对所有动作做评价：

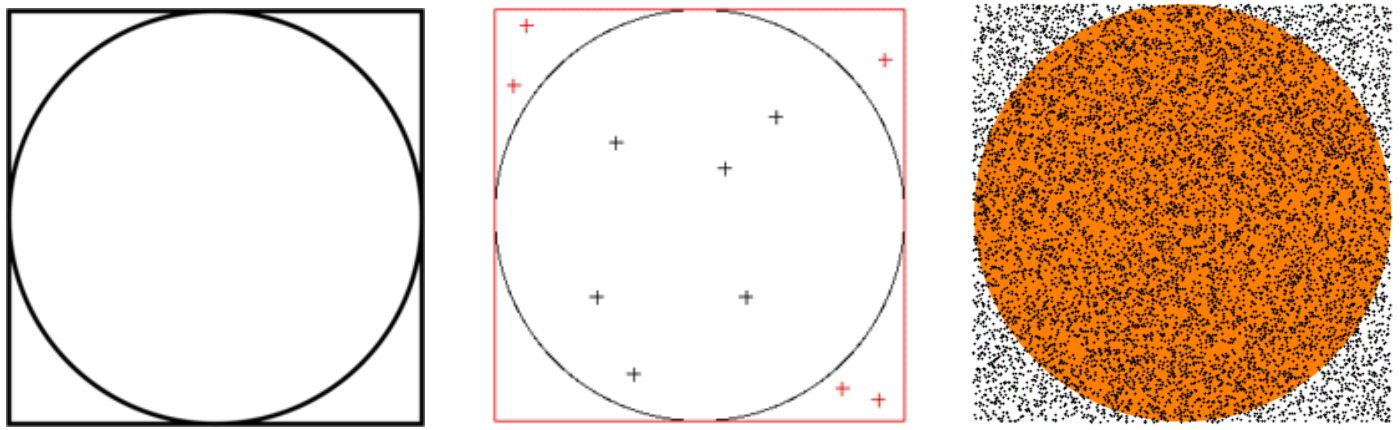
$$Q_\star(s_t, \text{左}) = 273, \qquad Q_\star(s_t, \text{右}) = -139, \qquad Q_\star(s_t, \text{上}) = 195$$

- 策略学习（Policy-based learning）通常是指学习最优策略函数$\pi^*(a|s)$，假设我们有了π^*，我们可以直接算出所有动作的概率值，然后去采样执行

$$\pi(\text{左} \,|\, s_t) = 0.6, \qquad \pi(\text{右} \,|\, s_t) = 0.1, \qquad \pi(\text{上} \,|\, s_t) = 0.3$$

蒙特卡洛方法

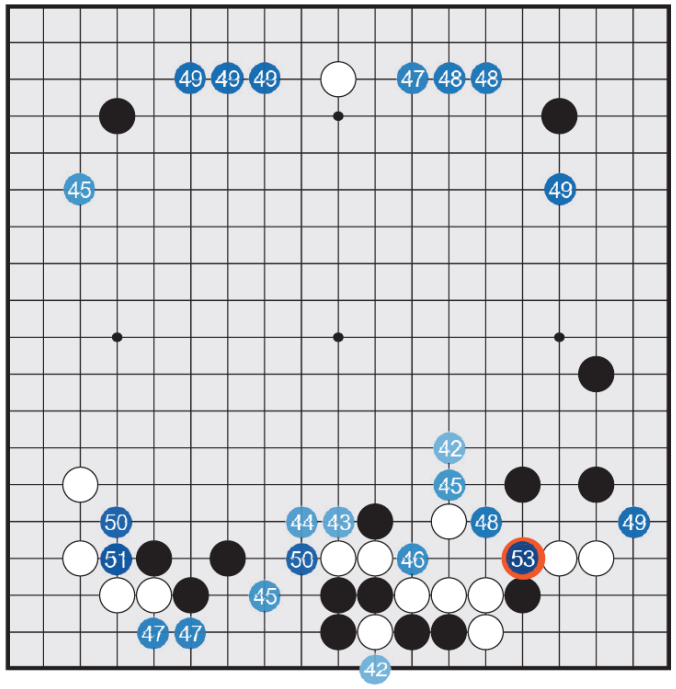
- 蒙特卡洛方法（Monte-Carlo methods）是一类广泛的计算算法。
  - 依赖于重复随机抽样来获得数值结果
- 例如，计算圆的面积

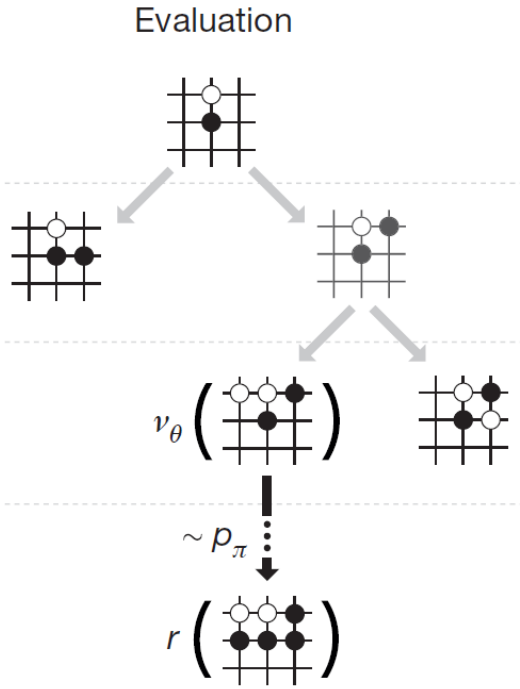


$$\text{Circle Surface} = \text{Square Surface} \times \frac{\#\text{points in circle}}{\#\text{points in total}}$$

蒙特卡洛方法

- 围棋对弈：估计当前状态下的胜率





$$\text{Win Rate}(s) = \frac{\#\text{win simulation cases started from } s}{\#\text{simulation cases started from } s \text{ in total}}$$

蒙特卡洛强化学习

- 在现实问题中，通常没有明确地给出状态转移和奖励函数，导致策略无法评估（模型未知导致无法做全概率展开）
- 在模型未知的情形下，从起始状态出发，使用某个策略进行采样，获得轨迹
  - Episode 1：$s_1, a_1, r_1, \ s_2, a_2, r_2, \ \cdots \ , \ s_T, a_T, r_T$
  - …
- 对于轨迹中出现的每一对状态-动作，记录其奖赏之和，作为该状态-动作对的一次累计奖赏采样值，多次采样得到多条轨迹之后，将每个状态-动作对的累计奖赏采样值进行平均，得到状态-动作值函数的估计

蒙特卡洛强化学习

- 在模型未知的情形下，从起始状态出发，使用某个策略进行采样，获得轨迹
  - Episode 1：$s_1, a_1, r_1,\ s_2, a_2, r_2,\ \cdots\ ,\ s_T, a_T, r_T$
  - …
- 如果策略是确定性的，对于某个状态只会输出一个动作，使用这样的策略进行采样，只能得到多条相同的轨迹
- 引入ϵ-贪心算法

$$\pi^{\epsilon}(s) = \begin{cases} \pi(s), & \text{以概率} 1-\epsilon \\ A\text{中以均匀概率随机选取动作} & \text{以概率} \epsilon \end{cases}$$

蒙特卡洛强化学习

- 在模型未知的情形下，从起始状态出发，使用某个策略进行采样，获得轨迹
  - Episode 1：$s_1, a_1, r_1,\ s_2, a_2, r_2,\ \cdots\ ,\ s_T, a_T, r_T$
  - …
- 在一个片段中的每个时间步长t的状态s都被访问
  - 增量计数器 $N(s) \leftarrow N(s) + 1$
  - 增量总累计奖励 $S(s) \leftarrow S(s) + G_t$
  - 价值被估计为累计奖励的均值 $V(s) = S(s)/N(s)$
- 同理，可计算出$Q(s, a)$

同策略蒙特卡洛学习

- 同策略(on-policy)被评估和改进的是同一个策略

循环{

- 根据策略π的ϵ贪心版本采样一条轨迹
- 更新$Q(s,a)$
- $$\pi(s)=\begin{cases}\arg\max_a Q(s,a) & \text{以概率}1-\epsilon\\ \text{以均匀概率从}A\text{中选取动作} & \text{以概率}\epsilon\end{cases}$$

}

异策略蒙特卡洛学习

- 能不能仅在评估时引入ϵ贪心策略，策略改进时改进原始策略

- 估计一个不同分布的期望

$$\begin{aligned}\mathbb{E}_{x\sim p}[f(x)] &= \int_x p(x)f(x)dx \\ &= \int_x q(x)\frac{p(x)}{q(x)}f(x)dx \\ &= \mathbb{E}_{x\sim q}\left[\frac{p(x)}{q(x)}f(x)\right]\end{aligned}$$

重要性采样

- 将每个实例的权重重新分配为 $\beta(x)=\frac{p(x)}{q(x)}$

异策略蒙特卡洛学习

- 能不能仅在评估时引入ϵ贪心策略，策略改进时改进原始策略

- 使用策略π的采样轨迹来评估策略π，实际上就是对累计奖赏估计期望

$$Q(s,a)=\frac{1}{m}\sum_{i=1}^{m}r_i$$

- 使用策略π'的采样轨迹来评估策略π，仅需对累计奖赏加权

$$Q(s,a)=\frac{1}{m}\sum_{i=1}^{m}\frac{p_i^{\pi}}{p_i^{\pi'}}r_i$$

p_i^{π}和$p_i^{\pi'}$分别表示两个策略产生第i条轨迹的概率

异策略蒙特卡洛学习

- 异策略(off-policy)被评估和改进的是不同的策略

循环{

- 执行π的ϵ贪心策略采样一条轨迹
- 以重要性加权的方式更新$Q(s,a)$
- $\pi(s) = \arg\max_a Q(s,a)$

}

增量蒙特卡洛更新

- 对于状态-动作对(s,a)，假定基于t个采样已经估计出值函数$Q_t^\pi(s,a)=\frac{1}{t}\sum_{i=1}^{t}G_i$
- 则在得到第$t+1$个采样时，有

$$Q_{t+1}^\pi(s,a)=Q_t^\pi(s,a)+\frac{1}{t+1}(G_{t+1}-Q_t^\pi(s,a))$$

- 更一般地，可以把$\frac{1}{t+1}$更换为系数a

$$Q_{t+1}^\pi(s,a)=Q_t^\pi(s,a)+\alpha\,(G_{t+1}-\ Q_t^\pi(s,a))$$

蒙特卡罗强化学习

- 蒙特卡罗强化学习通过考虑采样轨迹，克服了模型未知给策略估计造成的困难

- 蒙特卡罗强化学习的缺点：低效
  - 求平均时以“批处理式”进行
  - 在一个完整的采样轨迹完成后才对状态-动作值函数进行更新

- 克服缺点的办法：
  - 时序差分 (temporal difference，TD) 学习：同时结合了动态规划和蒙特卡罗方法的思想，能做到更高效的免模型学习

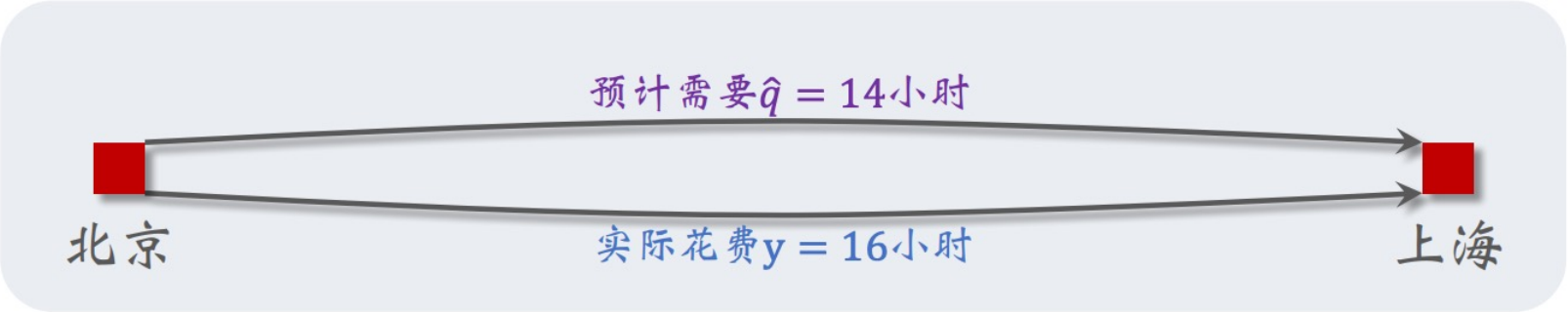
驾车时间预测的例子

- 假设我们有一个模型$Q(s,d)$，其中s是起点， d是终点，模型Q可以预测开车出行的时间开销，这个模型一开始不准确，但是得到更多数据之后，模型可以训练的越来越准

- 如何训练模型呢？在用户出发前，用户告诉模型起点和终点，模型做一个预测$\hat{q} = Q(s,d)$，当用户结束行程时，把实际用时y告诉模型，根据两者之差$y - \hat{q}$修正模型

驾车时间预测的例子

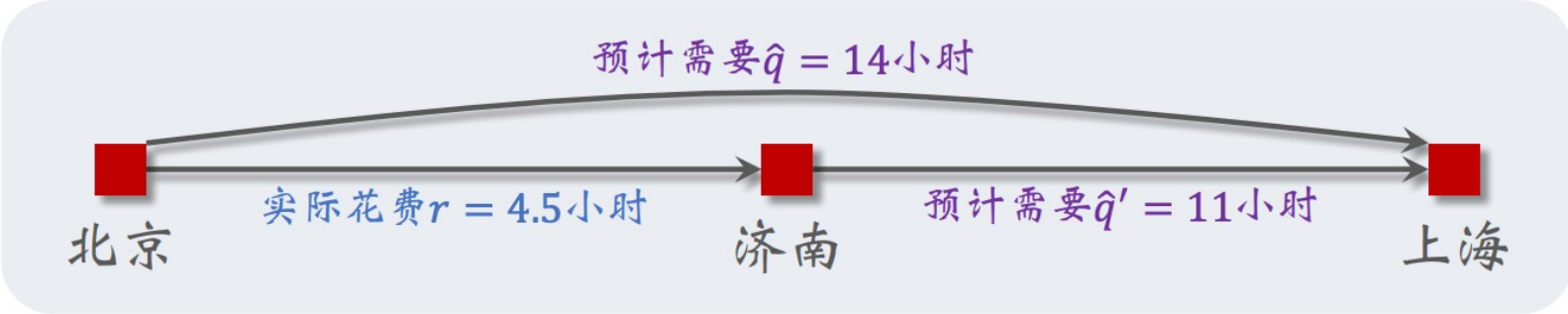
- 假设我要从北京开车去上海，出发前模型预测结果总车程为14小时，即$\hat{q} = 14$，到达上海时发现实际用时是16小时



- 基于两者之差（16-14）去更新模型，让模型预测更准

驾车时间预测的例子

- 出发前模型预测总车程为14小时，假设我过了4.5小时到达济南，让模型再做一次预测，模型告诉我从济南到上海需要11小时



- 根据模型的最新估计，整个旅程的总时间是15.5小时，$\hat{y} = r + \hat{q}' = 4.5 + 11 = 15.5$

- TD算法将$\hat{y} = 15.5$称为TD目标，它比最初的预测$\hat{q} = 14$更可靠

驾车时间预测的例子

- 因此，可以用$\hat{y} = 15.5$去对模型做修正，希望估计值$\hat{q}$尽量接近TD目标$\hat{y}$
- 基于两者之差（15.5-14）去更新模型
- 模型估计从北京到上海全程需要14小时，模型还估计从济南到上海全程需要11小时。相当于模型估计从北京到济南需要的时间为3小时
- 而实际花费4.5小时，模型估计与真实观测之差$\delta = 3 - 4.5 = -1.5$（TD误差）

时序差分学习

$$Q^{\pi}(s,a) = \sum_{s' \in S} P_{sa}(s')(r_{sa}(s') + \gamma V^{\pi}(s'))$$

$$= \sum_{s' \in S} P_{sa}(s')(r_{sa}(s') + \gamma \sum_{a'} \pi(s',a') Q^{\pi}(s',a'))$$

蒙特卡洛强化学习

$$Q_{t+1}^{\pi}(s,a) = Q_t^{\pi}(s,a) + \alpha\,(G_{t+1} - Q_t^{\pi}(s,a))$$

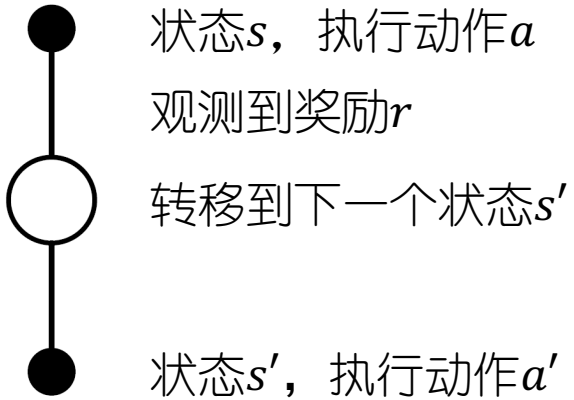
通过增量求和可以得到：

$$Q_{t+1}^{\pi}(s,a) = Q_t^{\pi}(s,a) + \alpha\big(r_{sa}(s') + \gamma Q_t^{\pi}(s',a') - Q_t^{\pi}(s,a)\big)$$

其中s'是前一次在状态s处执行动作a后转移到的状态， a'是策略π在s'选择的动作

SARSA

- 对于当前策略执行的每个（状态–动作–奖励–状态–动作）元组



- SARSA更新状态–动作值函数为

$$Q(s,a) \leftarrow Q(s,a) + \alpha(r_{sa}(s') + \gamma Q(s',a') - Q(s,a))$$

SARSA

Sarsa: An on-policy TD control algorithm

Initialize $Q(s,a), \forall s \in \mathcal{S}, a \in \mathcal{A}(s)$, arbitrarily, and $Q(\textit{terminal-state}, \cdot) = 0$
Repeat (for each episode):
  Initialize S
  Choose A from S using policy derived from Q (e.g., ϵ-greedy)
  Repeat (for each step of episode):
    Take action A, observe R, S'
    Choose A' from S' using policy derived from Q (e.g., ϵ-greedy)
    $Q(S,A) \leftarrow Q(S,A) + \alpha\big[R + \gamma Q(S',A') - Q(S,A)\big]$
    $S \leftarrow S'$; $A \leftarrow A'$;
  until S is terminal

SARSA算法中的两个“A”都是由当前策略选择的

Q-Learning

- 根据行为策略选择动作$a_t \sim \mu(\cdot|s_t)$
- 根据目标策略选择后续动作 $a'_{t+1} \sim \pi(\cdot|s_t)$
- $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha\left(r_{t+1} + \gamma Q(s_{t+1}, a'_{t+1}) - Q(s_t, a_t)\right)$

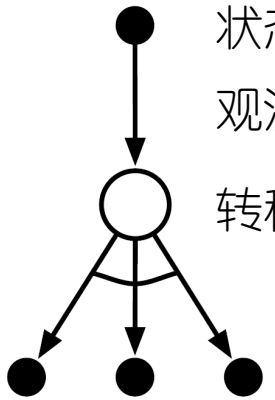
策略π的动作，而非策略μ

- 目标策略π是关于$Q(s, a)$的贪心策略

$$\pi(s_{t+1}) = \arg\max_{a'} Q(s_{t+1}, a')$$

Q-Learning

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha(r_{t+1} + \gamma \max_{a'} Q(s_{t+1}, a') - Q(s_t, a_t))$$



状态s，执行动作a

观测到奖励r

转移到下一状态s'

状态s'，执行动作 $\arg\max_{a'} Q(s', a')$

大纲

- 强化学习简介
- 多臂老虎机
- 有模型学习
- 无模型学习
- 深度强化学习

深度强化学习

AI = Deep Learning + Reinforcement Learning

Deep Reinforcement Learning

深度强化学习

$Q_\theta(s,a)$

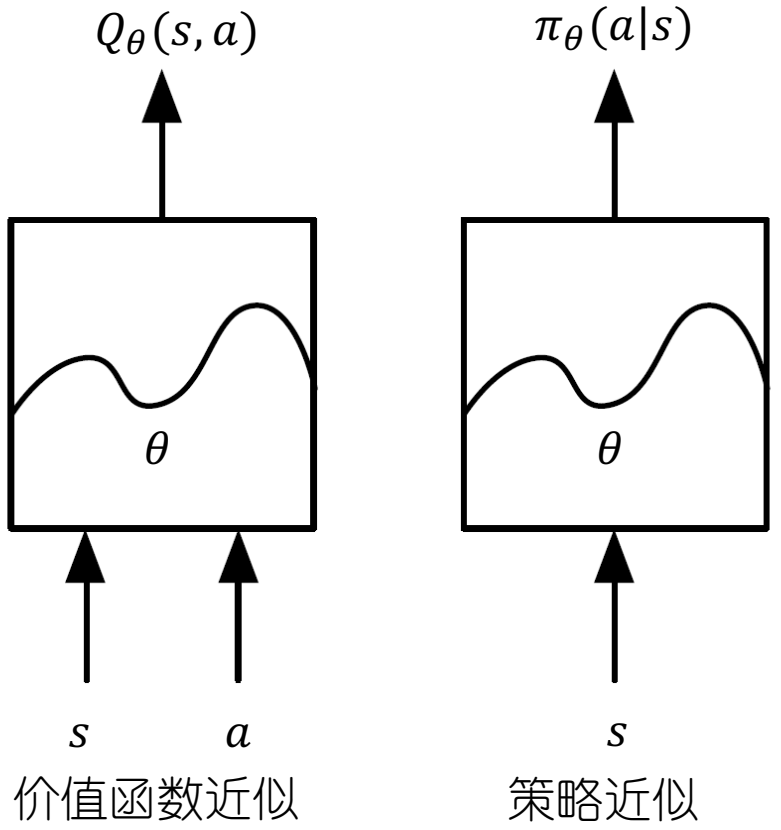
	a_1	a_2
s_1	3.2	4.1
s_2	0.8	1.2
s_3	2.3	-0.3

价值函数表格

$\pi_\theta(s,a)$

	a_1	a_2
s_1	0.2	0.8
s_2	0.4	0.6
s_3	0.9	0.1

策略表格



- 假如我们直接使用深度神经网络建立这些近似函数呢？
- 深度强化学习！

深度强化学习

- 2012年AlexNet在ImageNet比赛中大幅度领先对手获得冠军
- 2013年12月，第一篇深度强化学习论文出自NIPS 2013 Reinforcement Learning Workshop

Playing Atari with Deep Reinforcement Learning

Volodymyr Mnih **Koray Kavukcuoglu** **David Silver** **Alex Graves** **Ioannis Antonoglou**

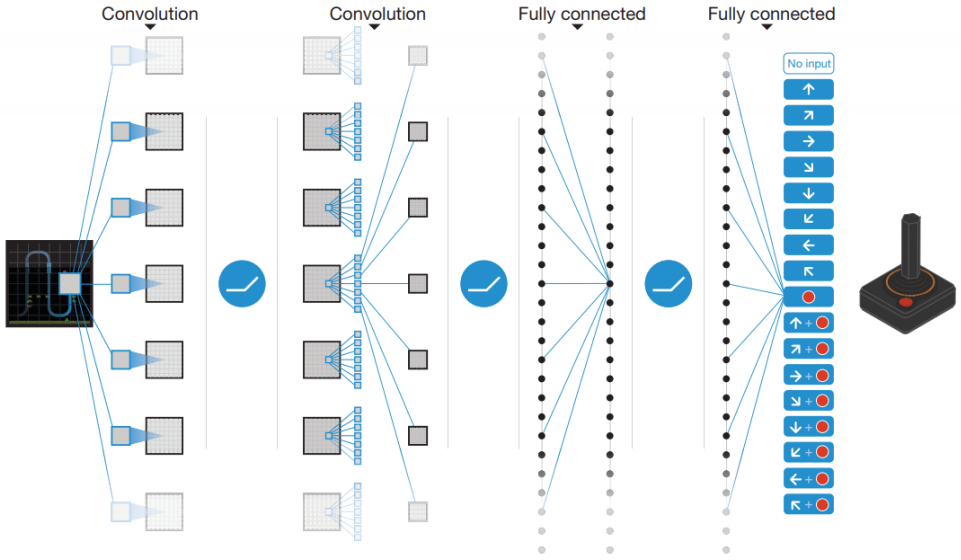
Daan Wierstra **Martin Riedmiller**

DeepMind Technologies

{vlad,koray,david,alex.graves,ioannis,daan,martin.riedmiller} @ deepmind.com

Volodymyr Mnih, Koray Kavukcuoglu, David Silver et al. Playing Atari with Deep Reinforcement Learning. NIPS 2013 workshop.

深度强化学习

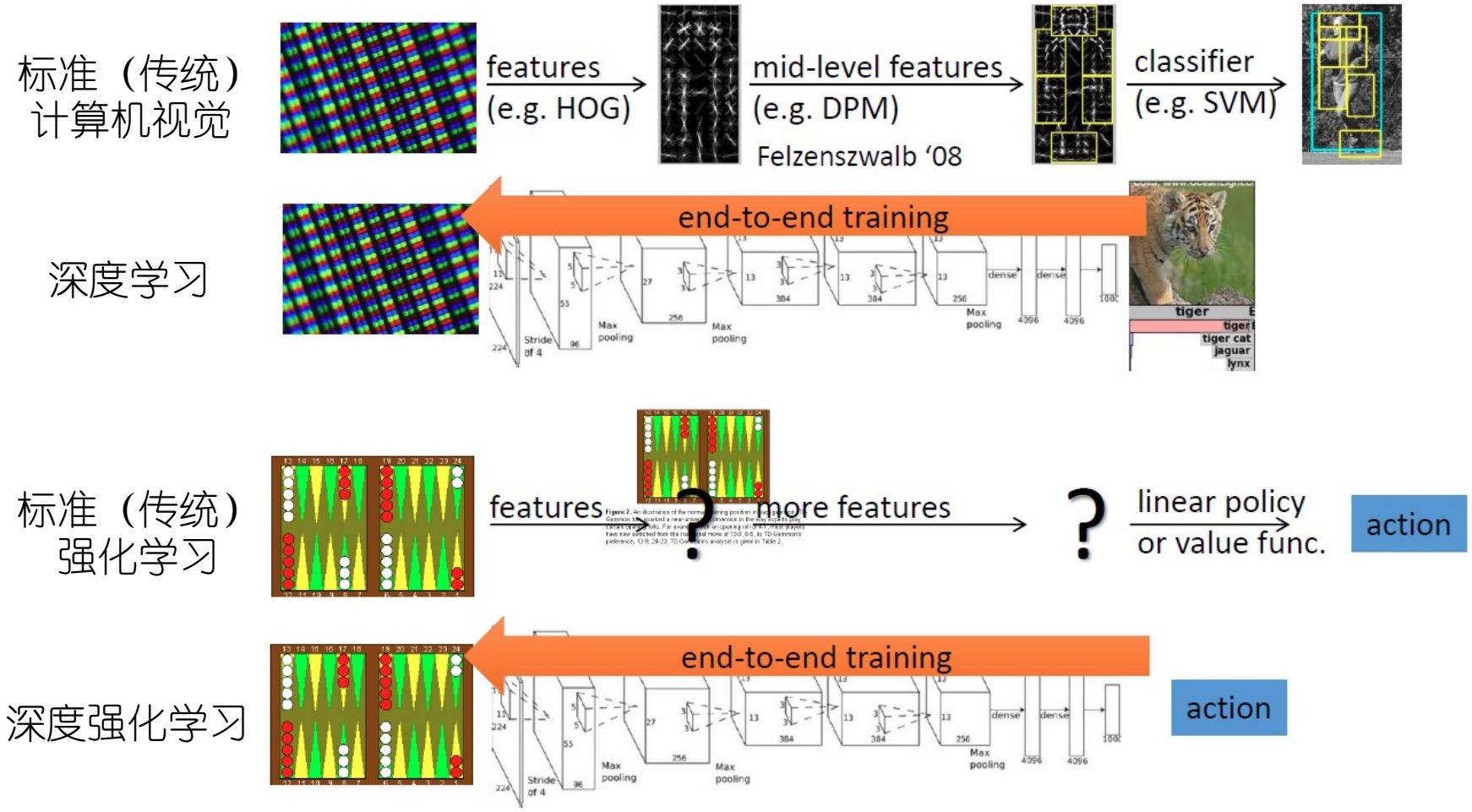


$$\nabla_{\theta_i} L_i(\theta_i) = \mathbb{E}_{s,a\sim\rho(\cdot);s'\sim\mathcal{E}}\left[\left(r + \gamma \max_{a'} Q(s', a'; \theta_{i-1}) - Q(s, a; \theta_i)\right) \nabla_{\theta_i} Q(s, a; \theta_i)\right]$$

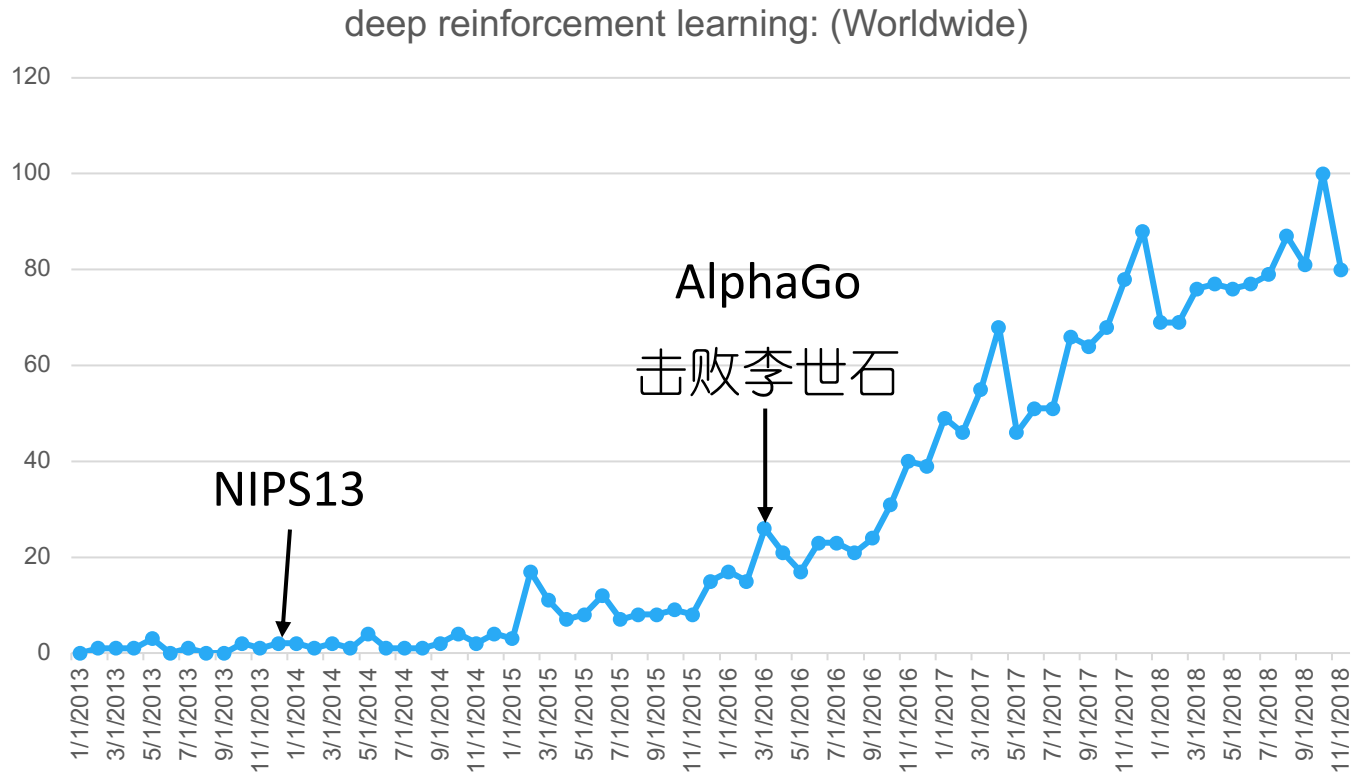
Q函数的参数通过神经网络反向传播学习

Volodymyr Mnih, Koray Kavukcuoglu, David Silver et al. Playing Atari with Deep Reinforcement Learning. NIPS 2013 workshop.

深度强化学习


标准（传统）
计算机视觉
features
(e.g. HOG)
mid-level features
(e.g. DPM)
Felzenszwalb '08
classifier
(e.g. SVM)
深度学习
end-to-end training
tiger
tiger cat
jaguar
lynx
标准（传统）
强化学习
features
more features
?
linear policy
or value func.
action
深度强化学习
end-to-end training
action

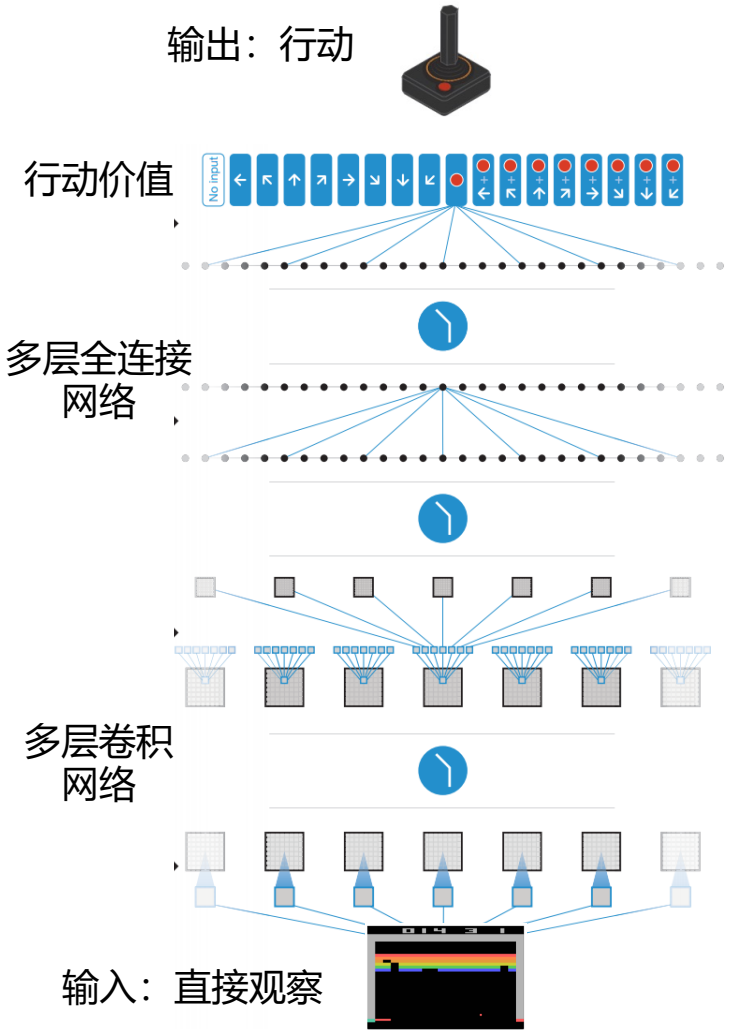
深度强化学习的趋势



Google搜索中词条“深度强化学习（deep reinforcement learning）”的趋势

深度强化学习

- 将深度学习（DL）和强化学习（RL）结合在一起会发生什么？
  - 价值函数和策略变成了深度神经网络
  - 相当高维的参数空间
  - 难以稳定地训练
  - 容易过拟合
  - 需要大量的数据
  - 需要高性能计算
  - CPU（用于收集经验数据）和GPU（用于训练神经网络）之间的平衡
  - …
- 这些新的问题促进着深度强化学习算法的创新



深度强化学习的研究前沿

基于模拟模型的强化学习

- 模拟器的无比重要性

目标策动的层次化强化学习

- 长程任务的中间目标是桥梁的基石

模仿学习

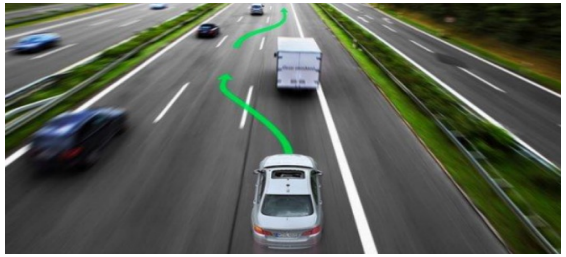
- 无奖励信号下跟随专家做策略学习

多智能体强化学习

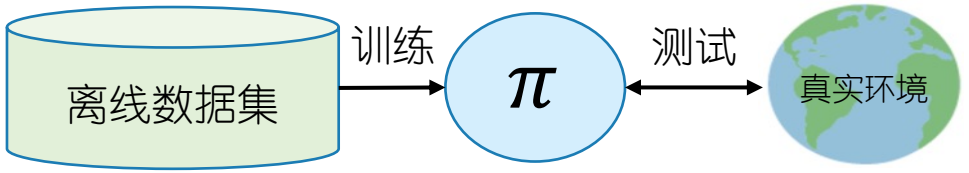
- 分散式、去中心化的人工智能

离线强化学习(offline RL)

- 动机：在真实环境中从零开始训练一个强化学习智能体往往不可取
  - 风险较高，例如无人驾驶归控、智能医疗等
  - 十分昂贵，例如机器人控制、推荐系统等



- 离线强化学习：在一个给定的离线数据集上直接训练出智能体策略，训练的过程中，智能体不得和环境做交互



模仿学习（Imitation Learning）

- 模仿学习：直接模仿人类专家的状态-动作对来学习策略
- 目的：都是学习策略，从而控制智能体
- 原理：
  - 强化学习利用环境反馈的奖励改进策略，目标是让累计奖励最大化
  - 模仿学习向人类专家学习，目标是让策略函数做出的决策与人类专家相同
- 不是真正的强化学习，而是强化学习的一种替代品

模仿学习算法

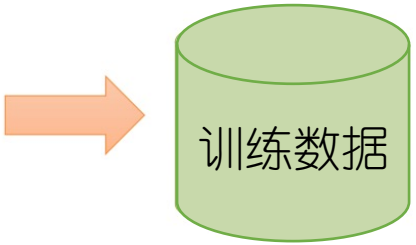
- 行为克隆(behavior cloning)或直接模仿学习
  - 无需奖励函数，模仿人类专家的动作来学习策略
- 逆强化学习(inverse reinforcement learning)
  - 找到能够使专家策略比任何其它策略更优的奖励函数
- 生成对抗模仿学习(Generative adversarial imitation learning)
  - 通过生成对抗网络的方式自动学习一个好的奖励函数

行为克隆

- 利用专家的决策轨迹，构造训练数据：状态作为特征，动作作为标记
- 利用数据集，使用分类/回归算法即可学得策略

$\mathbf{o}_t$

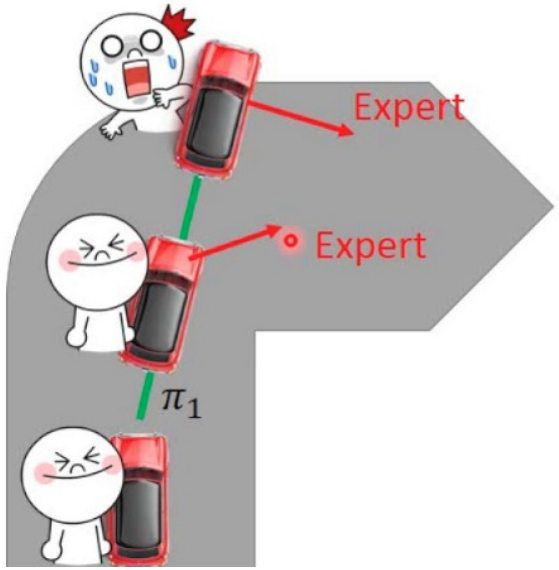
$\mathbf{a}_t$



$\pi_\theta(\mathbf{a}_t|\mathbf{o}_t)$

行为克隆

- 人类不会探索奇怪的状态和动作，因此数据集上的状态和动作缺乏多样性，
  智能体面对真实的环境，可能会遇到陌生的状态，做出的决策可能会很糟糕



行为克隆的优势在于离线训练，可以避免与真实环境交互，不会对环境造成影响（例如，行为克隆训练手术机器人）

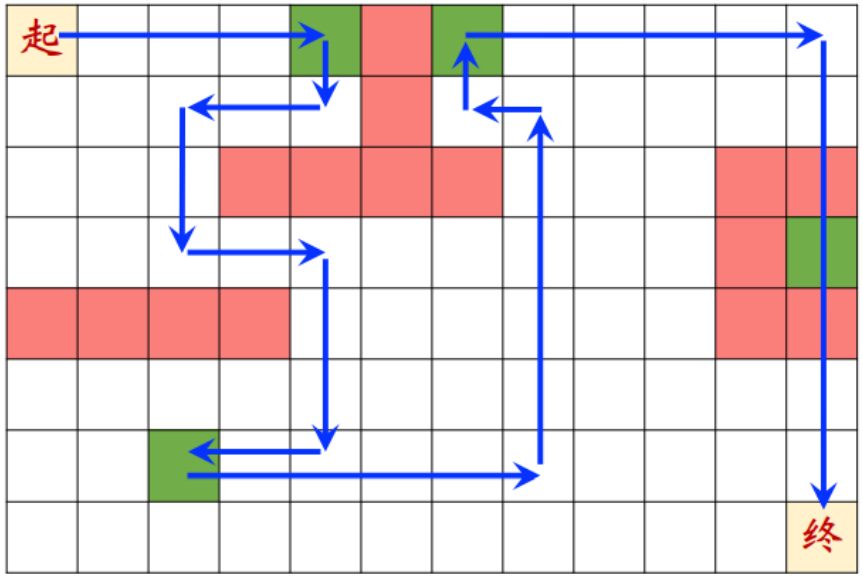
实践中，可以先通过行为克隆初始化策略函数，然后再进行强化学习

逆强化学习

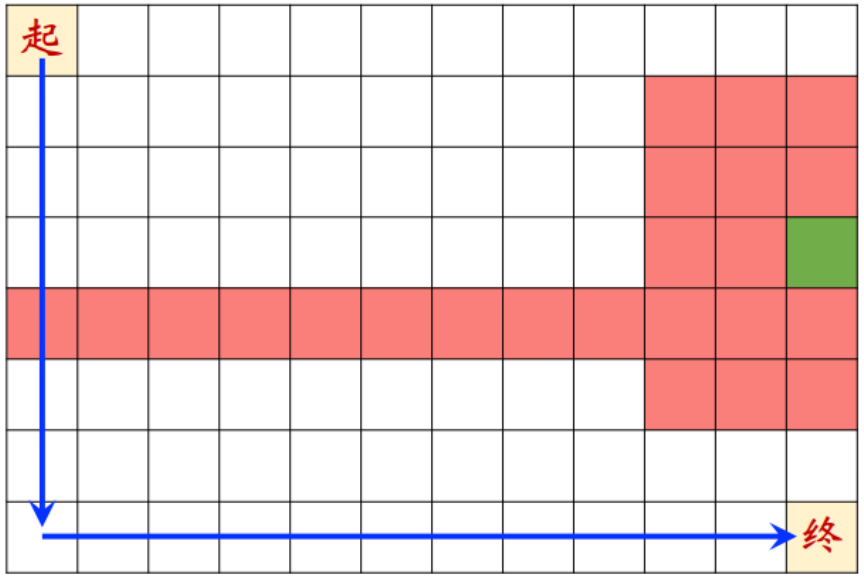
- 强化学习任务中，设计合理的符合应用场景的奖赏函数往往相当困难
- 缓解方法：从人类专家提供的范例数据中反推出奖赏函数
- 逆强化学习 (inverse reinforcement learning)
  - 基本思想：寻找某种奖赏函数使得范例数据是最优的，然后即可使用这个奖赏函数来训练策略

逆强化学习

- 蓝色箭头代表专家策略，如何反推奖励函数？



(a)

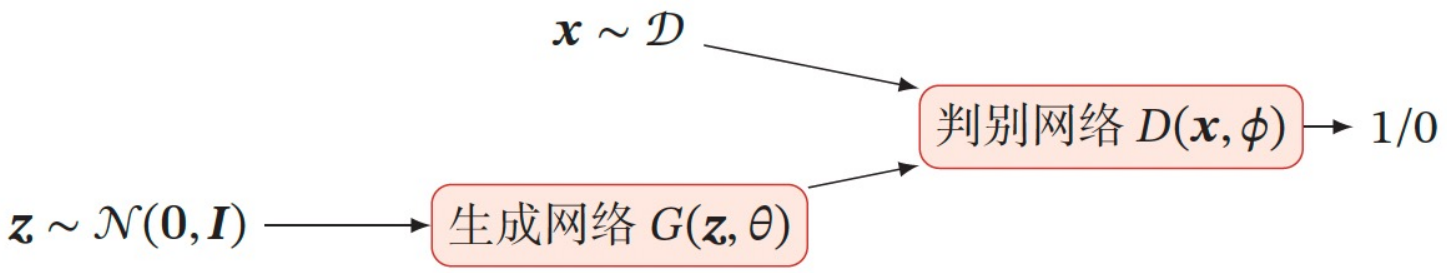


(b)

生成对抗模仿学习(Generative Adversarial imitation learning)

- 生成对抗网络[Goodfellow et al., 2014]通过对抗训练的方式使得生成网络产生的样本服从真实数据分布

- 判别网络(Discriminator)：目标是尽量准确地判断一个样本是来自于真实数据还是由生成网络产生
- 生成网络(Geneartor)：目标是尽量生成判别网络无法区分来源的样本



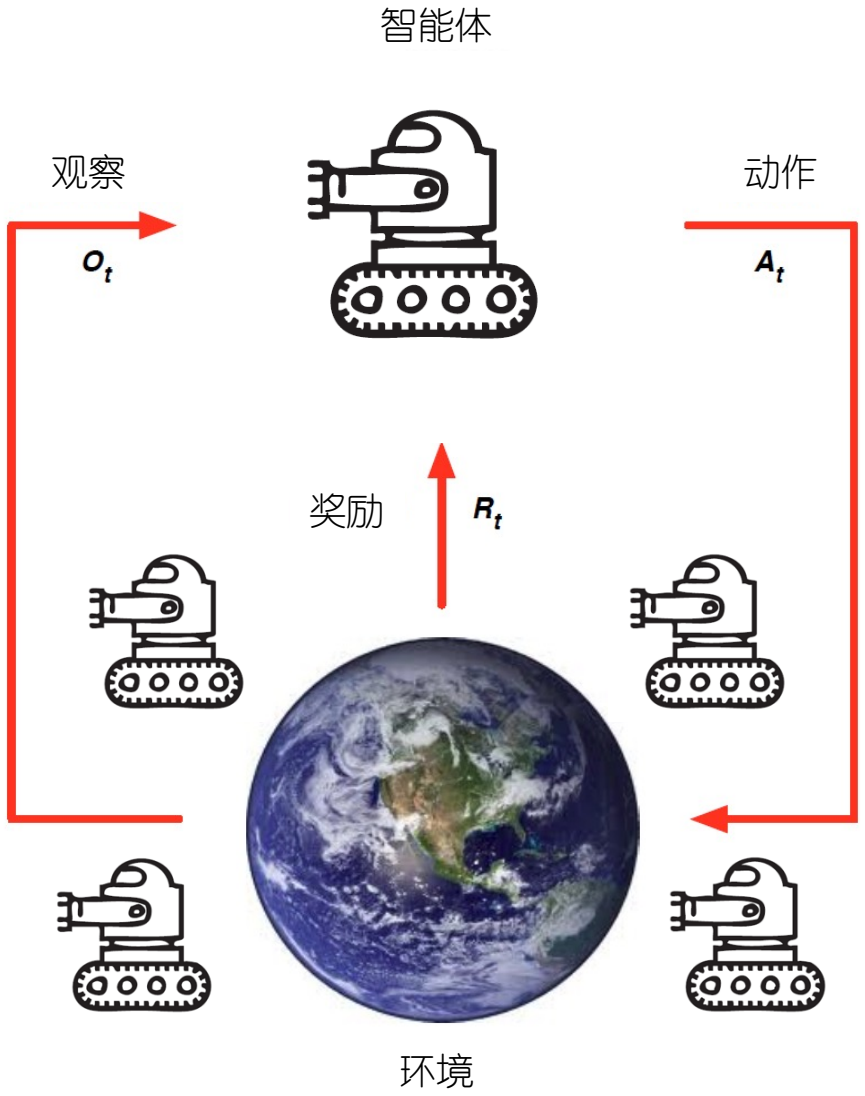
生成对抗模仿学习(Generative Adversarial imitation learning)

- 训练数据：人类专家操作智能体得到的轨迹数据
- 训练目标：让生成器生成的轨迹与数据集中的轨迹一样好，在训练结束的时候，判别器无法区分生成的轨迹与数据集里的轨迹

- 生成网络：输入状态s，输出选择各个动作的概率
- 判别网络：输入状态s，输出$p(s,a)$，越接近1表示动作a是人类专家做的，越接近于0表示动作a是生成网络生成的

多智能体强化学习

- 在与环境的交互过程中学习
- 环境包含有不断进行学习和更新的其他智能体
- 在任何一个智能体的视角下，环境是非稳态的（non-stationary）
  - 状态转移的概率分布会发生改变

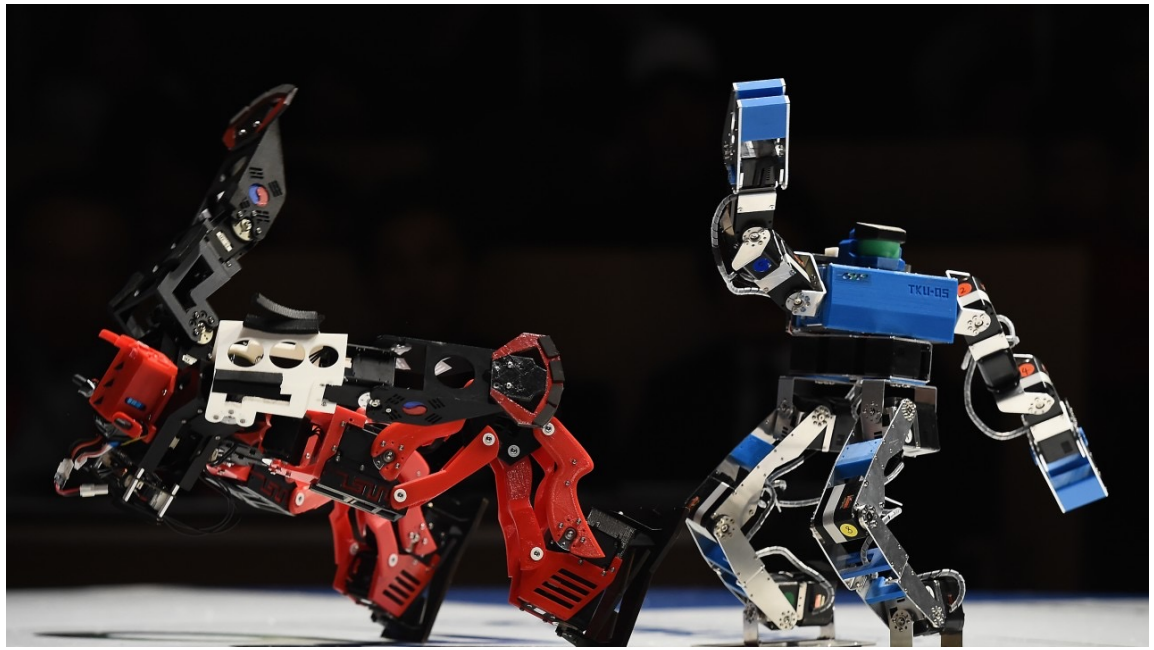


多智能体系统的常见设定

完全合作(fully cooperative)

多智能体系统的常见设定

完全竞争(fully competitive)

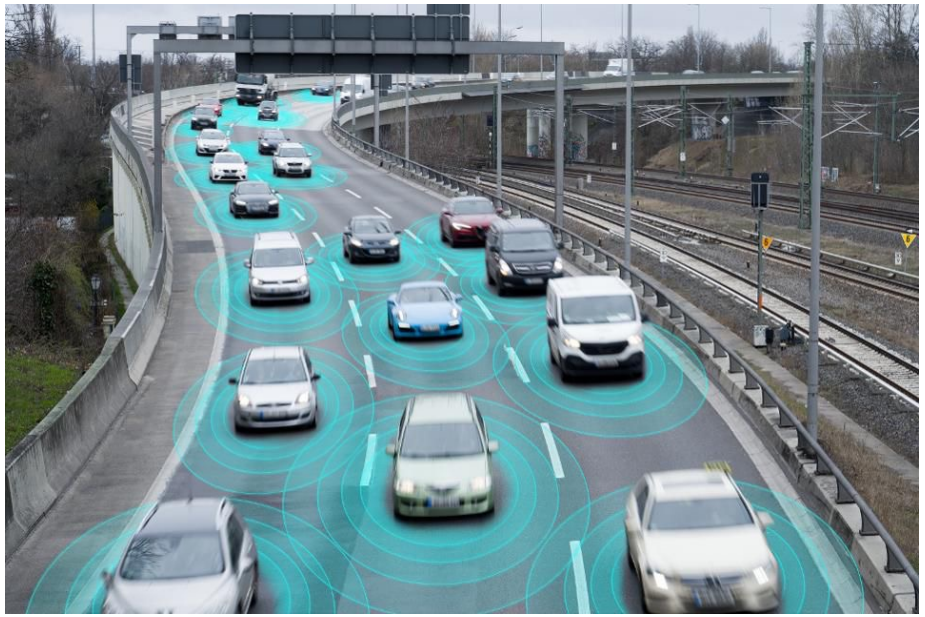


多智能体系统的常见设定

合作与竞争混合
(mixed cooperative & competitive)

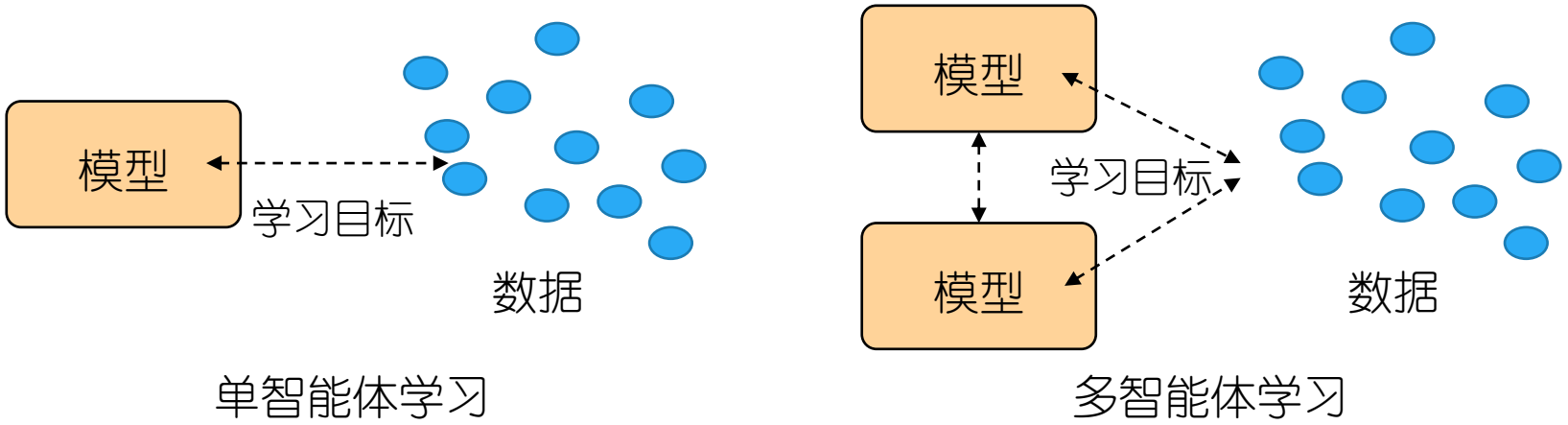
多智能体系统的常见设定

利己主义(self-interested)



多智能体强化学习的难点

- 多智能体学习从原理上来讲更加困难
  - 智能体不仅要与环境进行交互，还要相互之间进行交互



多智能体强化学习的分类

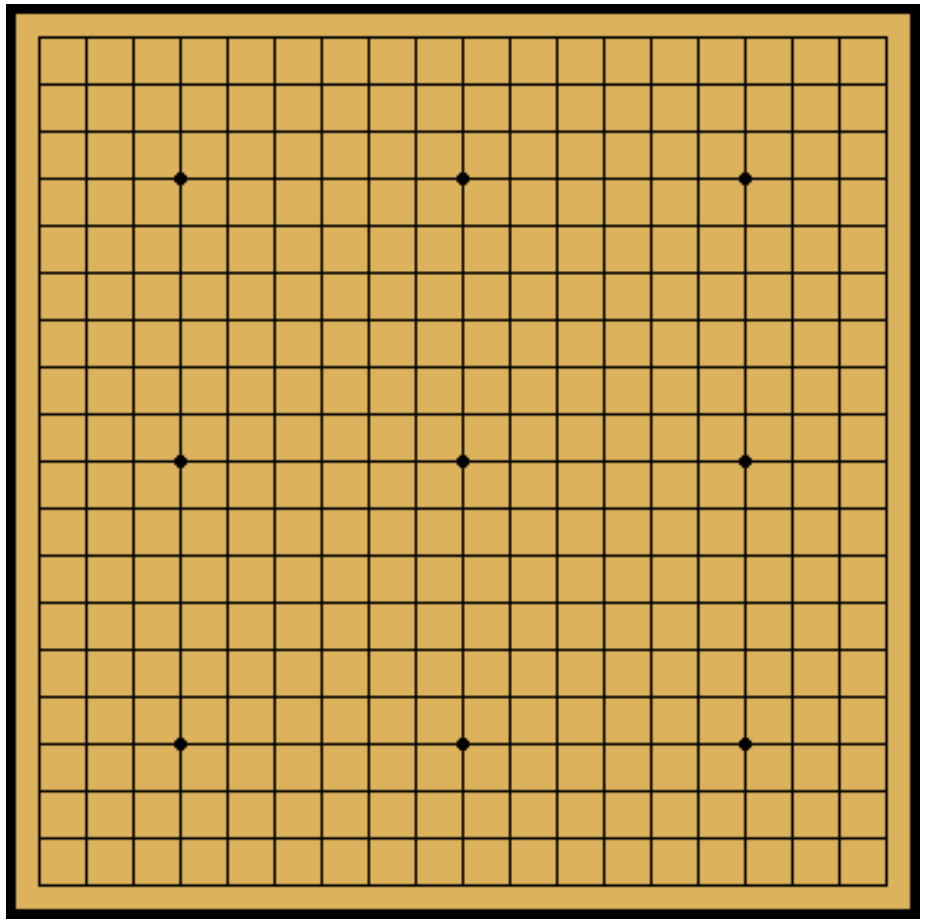
- 完全中心化方法（fully centralized）
  - 将多个智能体进行决策当作一个超级智能体在做决策，也即是把所有智能体的状态聚合在一起当作一个全局的超级状态，把所有智能体的动作连起来作为一个联合动作
  - 优点：环境是稳态；缺点：复杂度高
- 完全去中心化方法（fully decentralized）
  - 假设每个智能体都在自身的环境中独立地进行学习，不考虑其他智能体的改变。完全去中心化方法直接对每个智能体用一个单智能体强化学习算法来学习
  - 优点：简单好实现；缺点：环境非稳态，很可能不收敛
- 中心化训练去中心化执行 （centralized training with decentralized execution）
  - 在训练的时候使用一些单个智能体看不到的全局信息而达到更好的训练效果，而在执行时不使用这些信息，每个智能体完全根据自己的策略直接行动，以达到去中心化执行的效果
  - 特点：介于完全中心化方法和完全去中心化方法之间

强化学习的落地场景

- 无人驾驶
- 游戏AI
- 交通灯调度
- 网约车派单
- 组合优化
- 推荐搜索系统
- 数据中心节能优化
- 对话系统
- 机器人控制
- 路由选路
- 工业互联网场景
- …

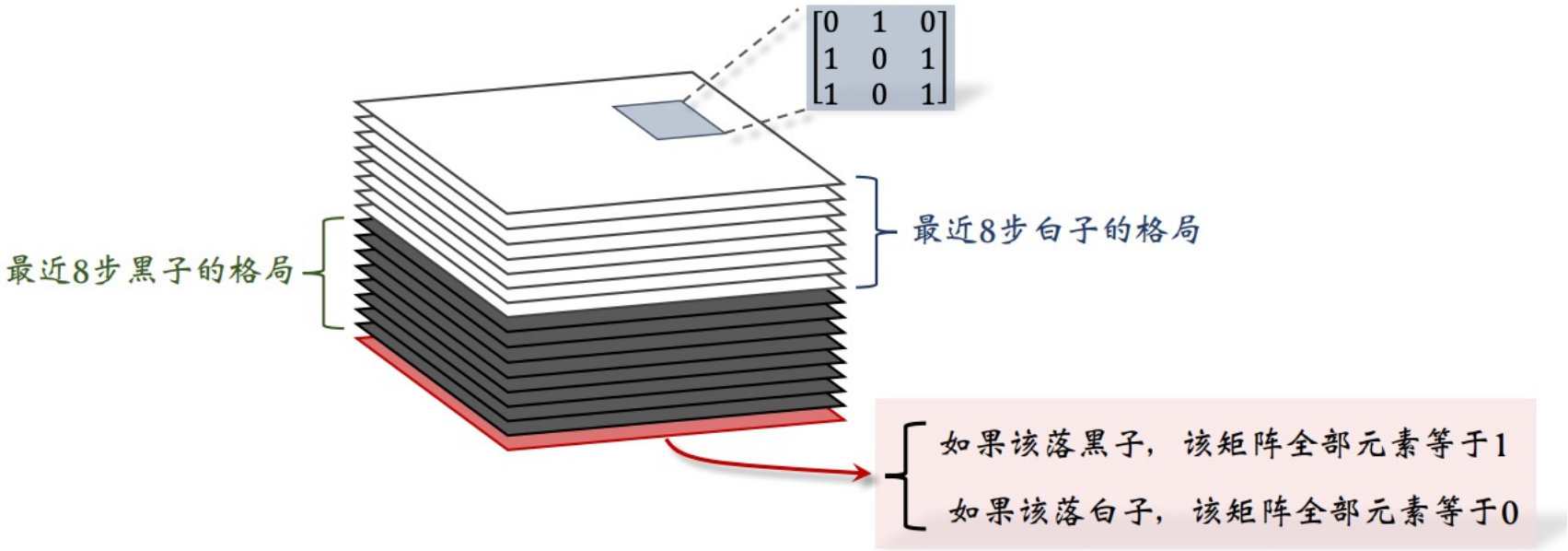
AlphaGO

围棋



- 围棋棋盘：$19\times19=361$
- State：19×19的矩阵
- Action: $\mathcal{A} \subset \{1, 2, 3, \cdots, 361\}$

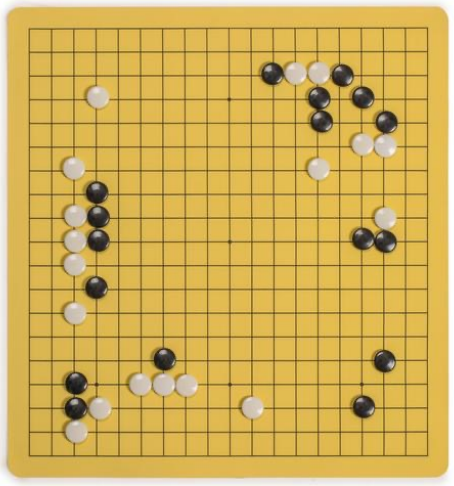
强化学习眼中的围棋



- AlphaZero使用的是19*19*17的张量表示一个状态
- 17：最近八步棋盘上黑子的位置，最近八步棋盘上白子的位置，以及当前该哪一方下棋（如果是黑棋，矩阵全为1，否则全为0）

策略网络(AlphaZero)

- 策略网络：$\pi(a|s;\theta)$，输入是状态，输出是每个动作的概率



state

19×19×17 tensor

Conv

feature

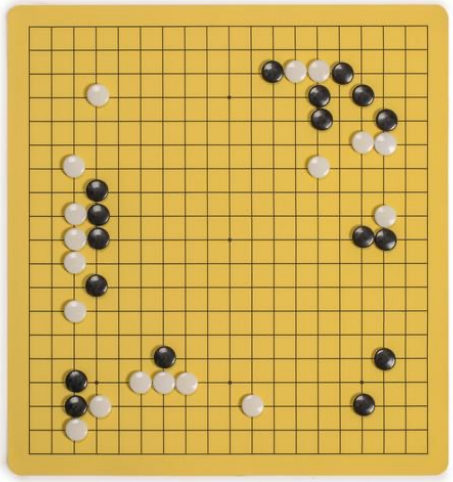
Dense

$\pi(1|s,\boldsymbol{\theta})$
$\pi(2|s,\boldsymbol{\theta})$
$\pi(3|s,\boldsymbol{\theta})$

$\pi(359|s,\boldsymbol{\theta})$
$\pi(360|s,\boldsymbol{\theta})$
$\pi(361|s,\boldsymbol{\theta})$

Probability distribution over the 361 actions

策略网络(AlphaGo)



state

19×19×48 tensor

Conv

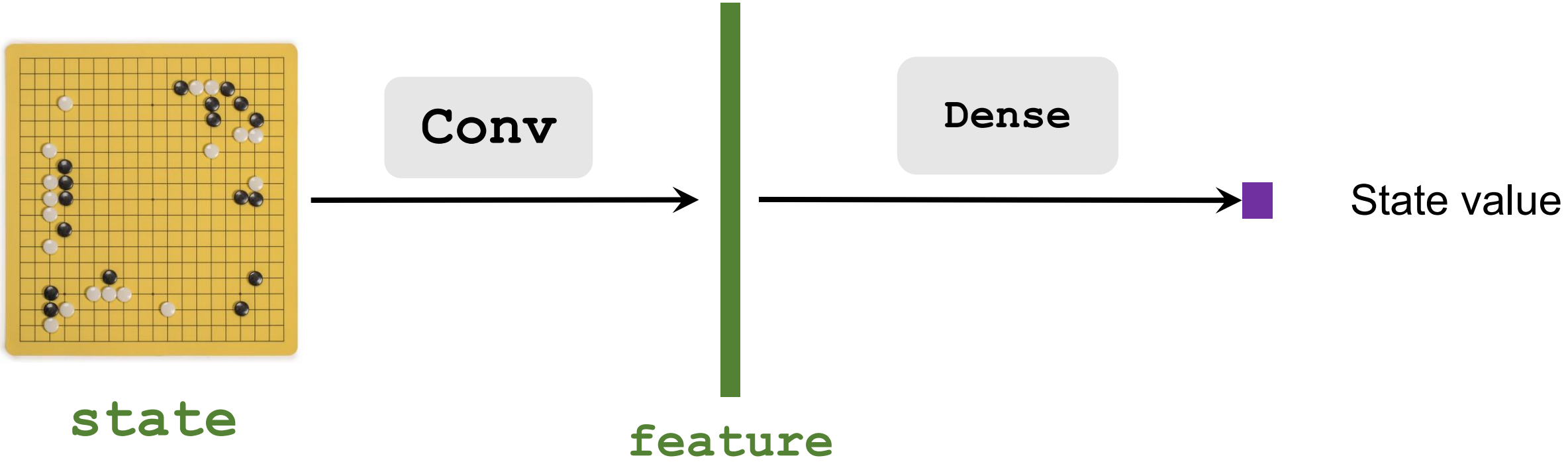
$\pi(1|s,\boldsymbol{\theta})$
$\pi(2|s,\boldsymbol{\theta})$
$\pi(3|s,\boldsymbol{\theta})$

$\pi(359|s,\boldsymbol{\theta})$
$\pi(360|s,\boldsymbol{\theta})$
$\pi(361|s,\boldsymbol{\theta})$

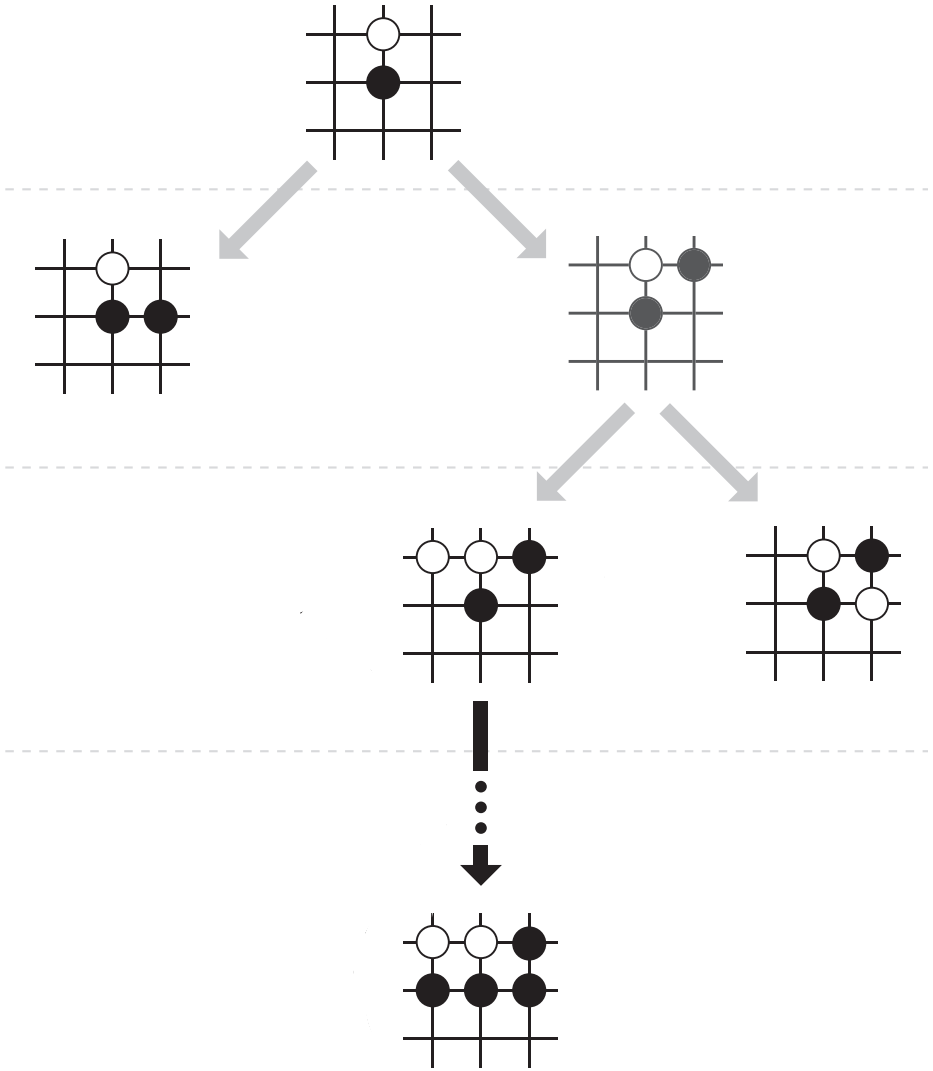
Probability distribution over the 361 actions

价值网络

- 价值网络：$v(s;w)$，对状态价值函数的估计，输入是状态，输出是一个实数，表示当前状态的好坏

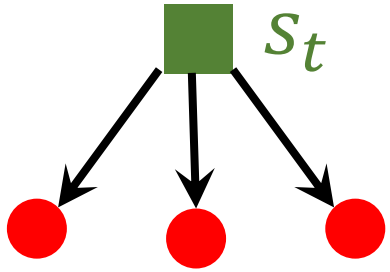


蒙特卡罗搜索



1. 选择(Selection)
2. 扩展(Expansion)
3. 评估(Evaluation)
4. 回溯(Backup)

第一步：选择

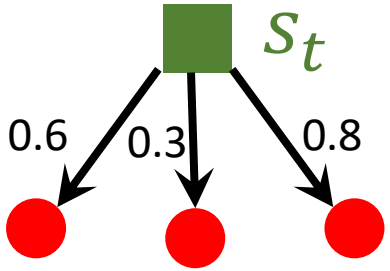


给定状态s_t，找出胜算较大的动作，只搜索这些好的动作

$$\text{score}(a) = Q(a) + \eta \cdot \frac{\pi(a \mid s_t; \boldsymbol{\theta})}{1 + N(a)}$$

- $Q(a)$：蒙特卡洛搜索模拟带来的动作价值估计
- $\pi(a \mid s_t; \theta)$：策略网络的输出
- $N(a)$：s_t下动作a已经被访问的次数

第一步：选择

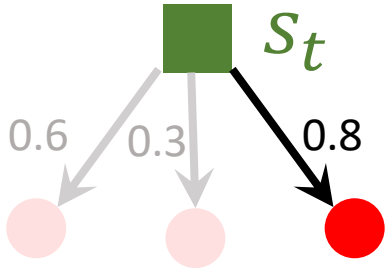


给定状态s_t，找出胜算较大的动作，只搜索这些好的动作

$$\text{score}(a) = Q(a) + \eta \cdot \frac{\pi(a \mid s_t; \boldsymbol{\theta})}{1 + N(a)}$$

- $Q(a)$：蒙特卡洛搜索模拟带来的动作价值估计
- $\pi(a \mid s_t; \theta)$：策略网络的输出
- $N(a)$：s_t下动作a已经被访问的次数

第一步：选择

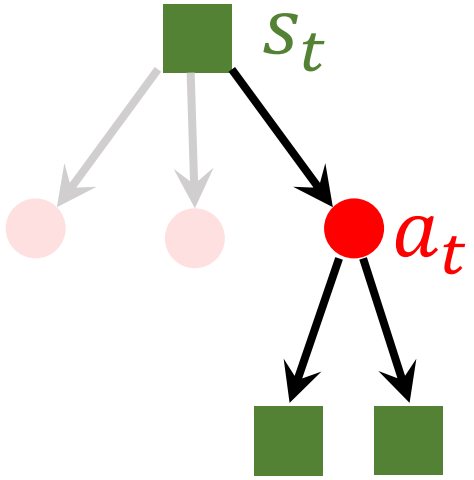


给定状态s_t，找出胜算较大的动作，只搜索这些好的动作

$$\text{score}(a) = Q(a) + \eta \cdot \frac{\pi(a \mid s_t; \boldsymbol{\theta})}{1 + N(a)}$$

- $Q(a)$：蒙特卡洛搜索模拟带来的动作价值估计
- $\pi(a \mid s_t; \theta)$：策略网络的输出
- $N(a)$：s_t下动作a已经被访问的次数

第二步：扩展

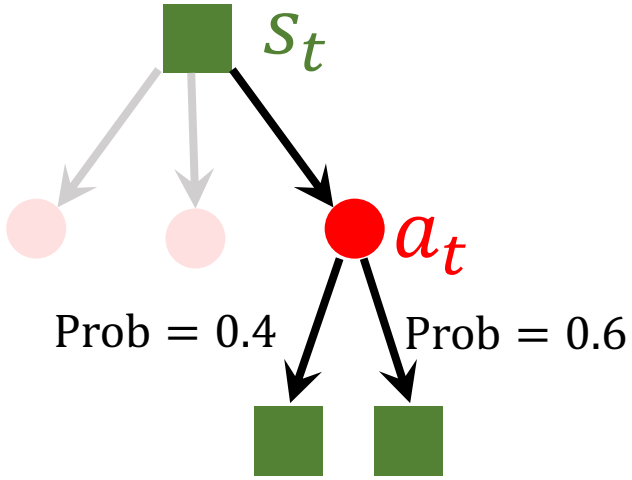


对手接下来会做什么动作？

AlphaGo用自己的策略网络去模拟对手，根据策略网络采样一个动作

$$a'_t \sim \pi(\cdot \mid s'_t; \boldsymbol{\theta})$$

第二步：扩展

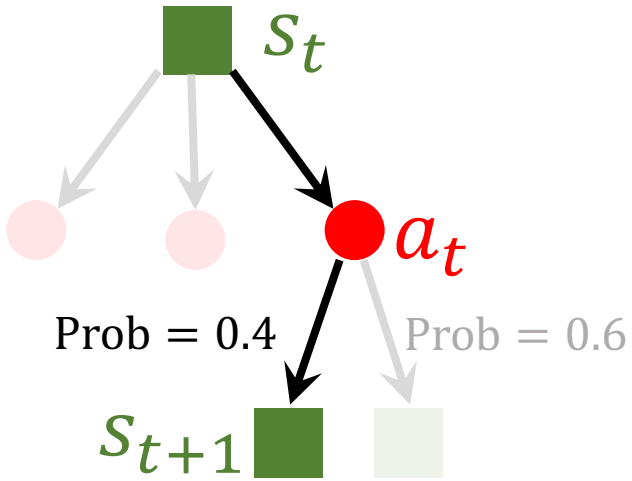


对手接下来会做什么动作？

AlphaGo用自己的策略网络去模拟对手，根据策略网络采样一个动作

$$a'_t \sim \pi(\cdot \mid s'_t; \boldsymbol{\theta})$$

第二步：扩展

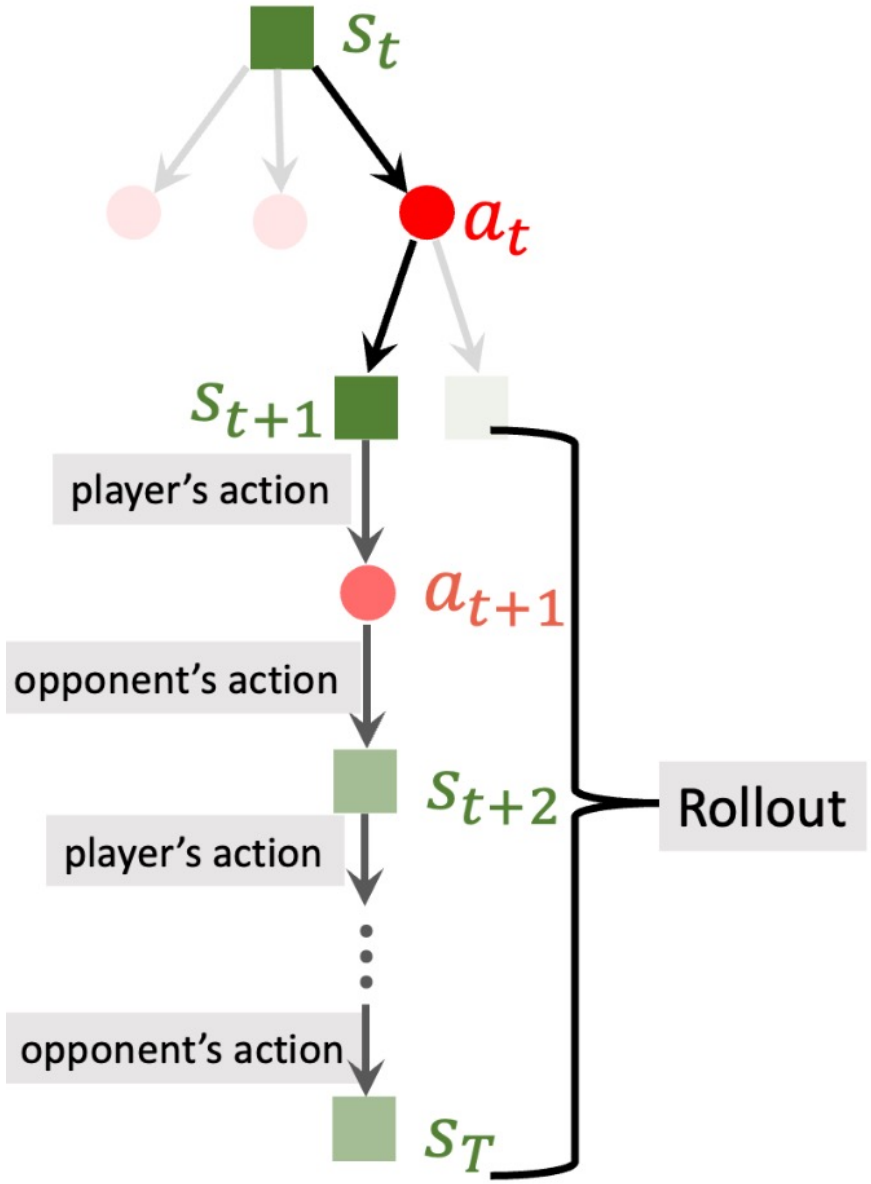


对手接下来会做什么动作？

AlphaGo用自己的策略网络去模拟对手，根据策略网络采样一个动作

$$a'_t \sim \pi(\cdot \mid s'_t; \boldsymbol{\theta})$$

第三步：模拟

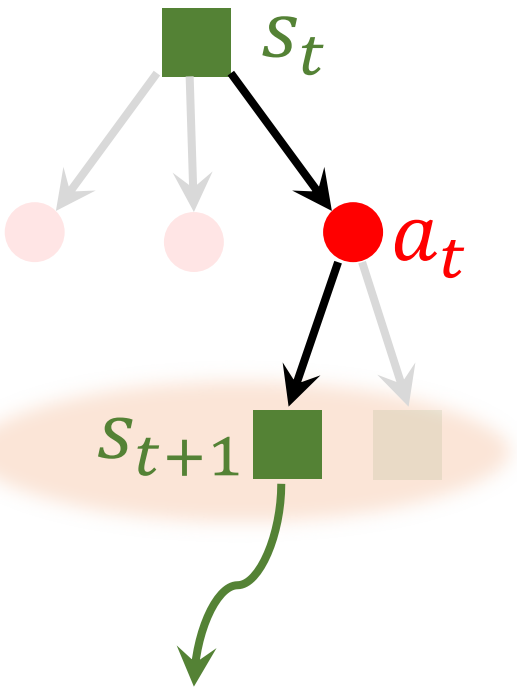


策略网络自我博弈，一直到分出胜负为止

- Player's action: $a_k \sim \pi(\cdot \mid s_k; \boldsymbol{\theta})$.
- Opponent's action: $a'_k \sim \pi(\cdot \mid s'_k; \boldsymbol{\theta})$.

- 游戏结束，获得奖赏r_T
  - Win: $r_T = +1$.
  - Lose: $r_T = -1$.

第三步：模拟



Record $V(s_{t+1})$

$$V(s_{t+1}) = \frac{1}{2} v(s_{t+1}; \mathbf{w}) + \frac{1}{2} r_T$$

策略网络自我博弈，一直到分出胜负为止

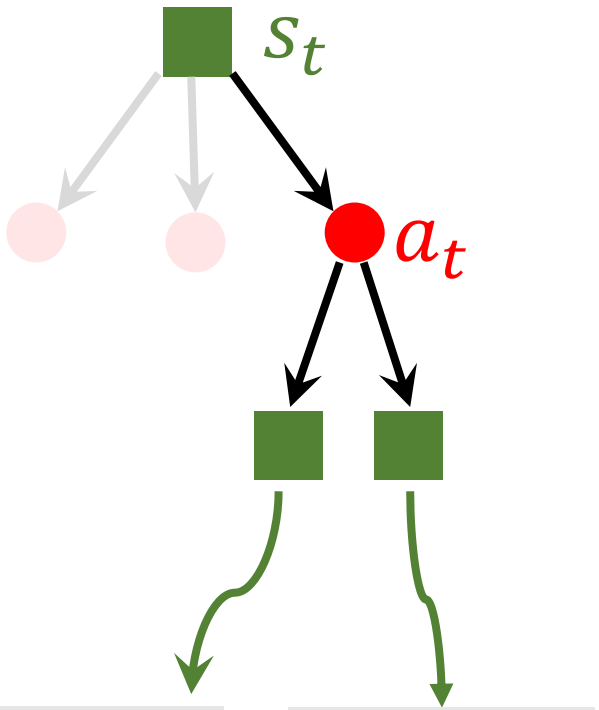
- Player's action: $a_k \sim \pi(\cdot \mid s_k; \boldsymbol{\theta})$.
- Opponent's action: $a'_k \sim \pi(\cdot \mid s'_k; \boldsymbol{\theta})$.

- 游戏结束，获得奖赏r_T
  - Win: $r_T = +1$.
  - Lose: $r_T = -1$.

- 此外，还可以通过价值网络评判s_{t+1}的好坏

- $v(s_{t+1}; \mathbf{w})$: output of the value network.

第四步：回溯



- 模拟过程会重复很多次，得到多个记录
- 把所有的记录做个平均，得到$Q(a_t)$

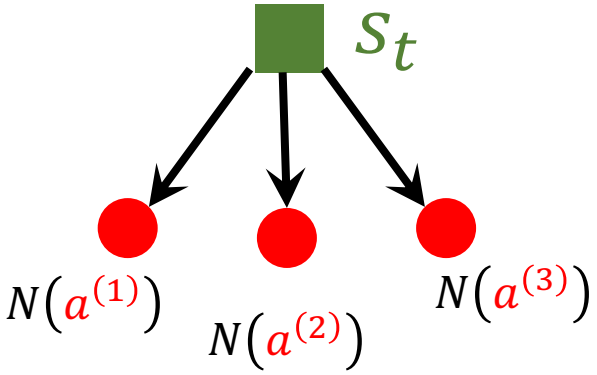
Records:

- $V_1^{(1)}$,
- $V_2^{(1)}$,
- $V_3^{(1)}$,
- $V_4^{(1)}$,
- ⋮

Records:

- $V_1^{(2)}$,
- $V_2^{(2)}$,
- $V_3^{(2)}$,
- $V_4^{(2)}$,
- ⋮

MCTS的决策



- $N(a)$: 记录每个动作在模拟过程中被选择的次数
- 真正的决策：

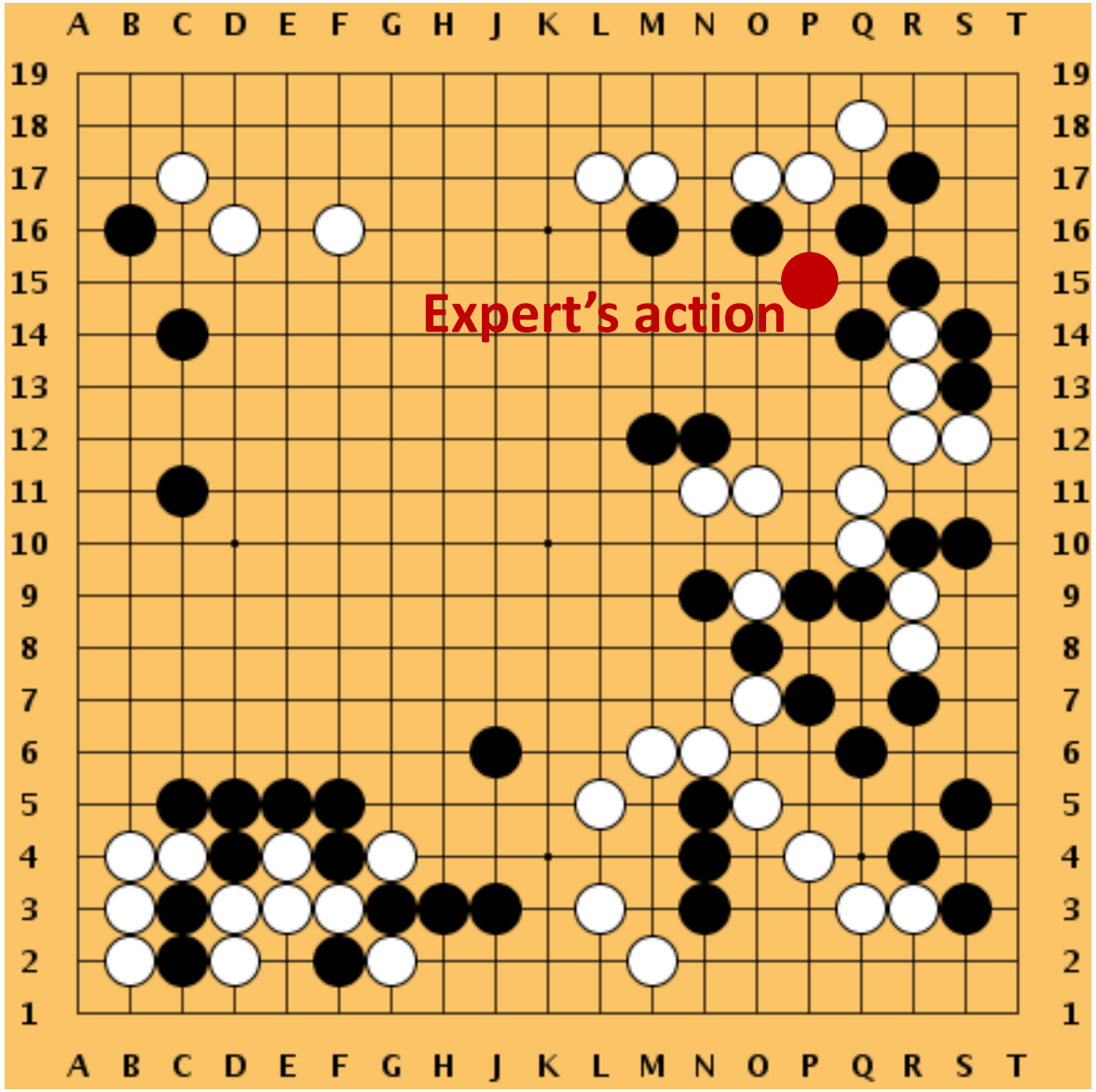
$$a_t = \underset{a}{\operatorname{argmax}}\, N(a)$$

策略网络的训练

- AlphaGo版本

1. 随机初始化策略网络，用行为克隆的方法从人类棋谱中学习策略网络
2. 让两个策略网络自我博弈，改进策略网络
3. 基于已经训练好的策略网络，训练价值网络

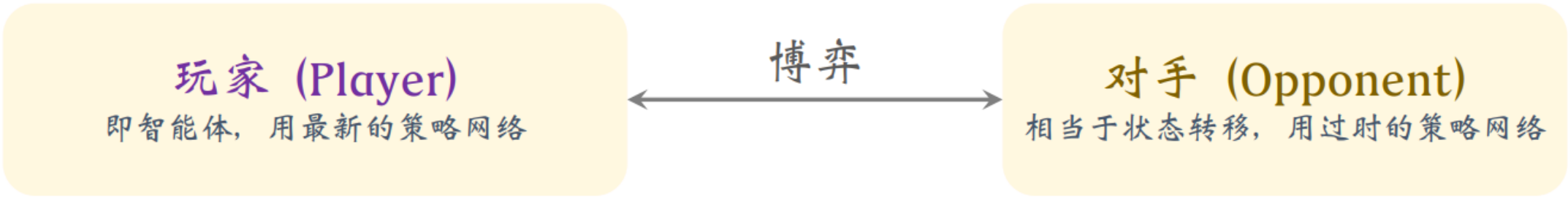
行为克隆



- 观察到状态 s_t
- 策略网络的预测
  $\mathbf{p}_t = [\pi(1|s_t, \boldsymbol{\theta}), \cdots, \pi(361|s_t, \boldsymbol{\theta})] \in (0,1)^{361}$
- 专家行动：$a_t^\star = 281$.
- $\mathbf{y}_t \in \{0,1\}^{361}$: one-hot vector of $a_t^\star = 281$.
- $\text{Loss} = \text{CrossEntropy}(\mathbf{y}_t, \mathbf{p}_t)$.
- 利用梯度下降更新策略网络

自我博弈

- 让策略网络做自我博弈，用胜负作为奖励，更新策略网络



训练价值网络

- 让训练好的策略网络做自我博弈，记录状态回报二元组 (s_t, g_t)
- 让价值网络$v(s_t; w)$去拟合回报g_t
- 回归问题

AlphaZero版本

- AlphaZero和AlphaGo2016版本最大的区别在于训练策略网络的方式，不再从人类棋谱学习，而是向MCTS学习
- 用MCTS控制两个玩家对弈，每走一步棋，MCTS需要做成千上万次模拟，并记录下每个动作被选中的次数$N(a)$
- 假设当前状态是s_t，执行MCTS，完成很多次模拟，得到361个整数：$N(1), \cdots, N(361)$，表示每个动作被选中的次数，归一化：

$$\boldsymbol{p}_t = \text{normalize}\left(\left[N(1),\ N(2),\ \cdots,\ N(361)\right]^T\right)$$

- 更新策略网络：让$\pi(\cdot\,|s_t, \theta)$尽量接近p_t

小结

- 强化学习：序列决策任务，延迟反馈、agent的动作会影响环境
- 有模型学习：MDP模型已知，动态规划问题
- 无模型学习：利用采样评估价值函数和价值动作函数，了解时序差分学习的基本思想
- 深度强化学习：利用神经网络建模值函数
- 了解模仿学习、多智能体强化学习、AlphaGo基本思想